UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/998,329	12/03/2001	Jeff L. Hunter	TI-33316	5547

23494 7590 02/09/2005

TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2122	

DATE MAILED: 02/09/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/998,329	HUNTER ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 December 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 December 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-19 are pending and have been examined. The priority date considered for the application is January 24, 2001.

Information Disclosure Statement

2. The listing of references in the specification (see, for example, the patents referenced on pages 7 and 11) is not a proper information disclosure statement. 37 CFR 1.98(b) requires a list of all patents, publications, or other information submitted for consideration by the Office, and MPEP § 609 A(1) states, "the list may not be incorporated into the specification but must be submitted in a separate paper." Therefore, unless the references have been cited by the examiner on form PTO-892, they have not been considered.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground

provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-19 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-24 of copending Application No. 09/998,756. Although the conflicting claims are not identical, they are not patentably distinct from each other because both recite analogous systems and methods for debugging a shared memory multiprocessor system and maintaining cache and breakpoint coherency with a software memory bus and a software memory map.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

5. Claims 1-19 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-13 of copending Application No. 09/998,755. Although the conflicting claims are not identical, they are not patentably distinct from each other because both recite analogous systems and methods for debugging a shared memory multiprocessor system and maintaining cache and breakpoint coherency with a software memory bus and a software memory map.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

6. Claims 1-19 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-10 of copending Application No. 09/998,330. Although the conflicting claims are not identical, they are not patentably distinct from each other because both recite analogous systems and methods for debugging a shared memory multiprocessor system and maintaining cache and breakpoint coherency with a software memory bus and a software memory map.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claim 12 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 12 recites the step of writing "the original instruction stored in a software representation maintained for software breakpoints." There is insufficient antecedent basis for this limitation in the claim. Claims 1, 7 and 9 do not recite storing an original instruction in a software representation maintained for software breakpoints.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-5, 7, 8 and 13-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Pat. No. 6,065,078 to Falik et al. ("Falik") in view of U.S. Pat. No. 6,088,770 to Tarui et al. ("Tarui").

With respect to claim 1, Falik discloses a software development system for debugging software on a target system having a plurality of processors (see, for example, column 1, lines 34-49) configured with shared memory (see, for example, column 17, lines 46-48).

Although Falik discloses an interface module for defining the processor configuration of the target system (see, for example, column 3, lines 9-16), and a system reset or initialization process (see, for example, column 3, lines 57-61), Falik does not expressly disclose:

(a) a setup utility that receives user input to define a hardware configuration of the target system; and

(b) an initialization process that receives the hardware configuration information from the setup utility and creates a software memory map of the target system.

However, Tarui discloses hardware configuration information that is defined for the target system (see, for example, column 9, lines 12-31), as in part (a) above, and an initialization

process that receives the hardware configuration information to establish a memory map of the local and shared memory (see, for example, column 6, lines 30-37), as in part (b) above. As in Falik, the target system of Tarui is similarly a shared memory multiprocessor system (see, for example, column 6, lines 5-8). The memory map is used, for example, to determine which processor controls the memory sought by an access request on the shared memory bus (see, for example, column 9, lines 1-11).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the software development system of Falik with the shared memory features taught by Tarui. One of ordinary skill in the art would have been motivated to provide a memory map, such as taught by Tarui, so as to determine which of the plurality of processors disclosed by Falik controls which range of memory.

Falik also discloses that the initialization process loads drivers for each of the plurality of processors, activates a first debug session associated with a first processor of the plurality of processors and activates at least a second debug session associated with a second processor of the plurality of processors (see, for example, FIG. 18, which shows a driver for each of the processors, and column 2, lines 37-43, which further shows a separate debugger or debug session for each processor), wherein each debug session is operable to transmit read requests and write requests to its associated processor (see, for example, column 4, lines 30-37, which shows that each debugger can transmit messages or requests to its associated processor).

Falik also discloses:

(c) a software memory bus that performs processing of shared memory access requests (see, for example, column 2, line 62 to column 3, line 3, which shows an interface or bus for processing shared resource and shared memory access requests).

With respect to claim 2, Falik does not expressly disclose the limitation in which the software memory bus performs processing of shared memory access requests using a method for transparently writing to shared memory when debugging a multiple processor system.

However, Tarui further discloses a method for transparently writing to shared memory (see, for example, the abstract). The method is used, for example, to reduce traffic and improve access latency on the bus (see, for example, column 19, lines 12-14).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the bus of Falik with the shared memory transparency features taught by Tarui. One of ordinary skill in the art would have been motivated to provide a method for transparently writing to shared memory, such as taught by Tarui, so as to reduce traffic and improve access latency on the bus disclosed by Falik.

Tarui further discloses:

(a) detecting a write request to a shared memory location by the first debug session (see, for example, column 9, lines 1-11, which shows detecting an access request to a shared memory location, and column 8, lines 7-11, which shows that such requests include write requests);

(b) if the first processor associated with the first debug session has write access to the shared memory location then selecting the first processor to perform the write request (see, for example, column 9, lines 42-47, which shows determining the processor that has access to the

shared memory location, and column 14, lines 22-26, which then shows selecting the first processor to perform the write request), else performing the following steps:

(i) searching the software memory map to determine if the second processor has write access to the shared memory location (see, for example, column 9, lines 32-41, which shows determining if a "remote" or second processor has access to the shared memory location based on the memory map);

(ii) selecting the second processor to perform the write request (see, for example, column 15, lines 46-52, which shows selecting the second processor to perform the write request); and

(c) passing the write request initiated by the first debug session to the selected processor for execution (see, for example, column 14, lines 22-26 and column 15, lines 46-52, which shows passing the write request to the selected processor).

With respect to claim 3, Tarui further discloses the limitation wherein the step of passing the write request comprises the steps of:

(a) searching the software memory map for a second plurality of processors (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location);

(b) broadcasting the write request to the second plurality of processors (see, for example, column 10, line 66 to column 11, line 5, which shows broadcasting the request to the plurality of processors); and

(c) performing cache coherency updates in response to the write request in each of the second plurality of processors (see, for example, column 6, lines 46-53, which shows issuing and

executing "CCC" commands, and column 3, lines 5-11, which shows that the CCC commands are for performing cache coherency updates).

With respect to claim 4, Tarui further discloses the limitation wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request (see, for example, column 14, line 5-14, which shows broadcasting an "I" command, and column 8, lines 7-11, which shows that the I command is for maintaining cache coherency as opposed to a normal write request).

With respect to claim 5, Tarui further discloses the limitation wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency (see, for example, column 8, lines 7-11, which shows using a cache coherency capability in response to a write request).

With respect to claim 7, Falik also discloses the limitation in which the software memory bus performs processing of shared memory access requests using a method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system (see, for example, column 16, lines 36-54), the method comprising the steps of:

(a) setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint (see, for example, column 16, lines 40-50, which shows setting a breakpoint such that the monitor or debug session of each processor is notified); and

(b) clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint (see, for

example, column 17, lines 37-48, which shows clearing the breakpoint such that the monitor or debug session of each processor is notified).

With respect to claim 8, Tarui further discloses the limitation wherein the step of setting comprises:

(a) searching the software memory map to find a first plurality of processors having read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location).

Falik also discloses:

(b) updating a software representation maintained for software breakpoints for each of the first plurality of processors (see, for example, column 16, lines 40-50, which shows updating the bits maintained for the breakpoints); and

(c) writing the software breakpoint instruction in the shared memory location (see, for example, column 16, lines 40-50, which shows writing the breakpoint to a shared location).

With respect to claim 13, Falik does not expressly disclose the limitation in which the software memory bus performs processing of shared memory access requests using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory.

However, Tarui further discloses a method for transparently maintaining cache coherency (see, for example, the abstract). The method is used, for example, to reduce traffic and improve access latency on the bus (see, for example, column 19, lines 12-14).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the bus of Falik with the shared memory transparency features taught by Tarui. One of ordinary skill in the art would have been motivated to provide a method for transparently maintaining cache coherency, such as taught by Tarui, so as to reduce traffic and improve access latency on the bus disclosed by Falik.

Tarui further discloses:

(a) denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache (see, for example, column 6, line 66 to column 7, line 9, which shows denoting in a memory table or map whether a "CCC" command is necessary for each processor, and column 3, lines 5-11, which shows that the CCC command applies to processors that have caches);

(b) detecting a write request to a shared memory location by the first debug session (see, for example, column 9, lines 1-11, which shows detecting an access request to a shared memory location, and column 8, lines 7-11, which shows that such requests include write requests);

(c) passing the write request initiated by the first debug session to the first processor for execution (see, for example, column 14, lines 22-26, which shows passing the write request to the first processor);

(d) searching the software representation of the memory map for a first plurality of processors that have read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location);

(e) broadcasting the write request to the first plurality of processors (see, for example, column 10, line 66 to column 11, line 5, which shows broadcasting the request to the plurality of processors); and

(f) performing cache coherency updates in response to the write request in each of the first plurality of processors (see, for example, column 6, lines 46-53, which shows issuing and executing "CCC" commands, and column 3, lines 5-11, which shows that the CCC commands are for performing cache coherency updates).

With respect to claim 14, Tarui further discloses the limitation wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request (see, for example, column 14, line 5-14, which shows broadcasting an "I" command, and column 8, lines 7-11, which shows that the I command is for maintaining cache coherency as opposed to a normal write request).

With respect to claim 15, Tarui further discloses the limitation wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency (see, for example, column 8, lines 7-11, which shows using a cache coherency capability in response to a write request).

With respect to claim 16, Tarui further discloses the limitation in which the software memory bus performs processing of shared memory access requests using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared instruction memory, the method comprising the steps of:

(a) denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache (see, for example, column 6, line 66 to column 7, line 9, which shows denoting in a memory table or map whether a "CCC" command is necessary for each processor, and column 3, lines 5-11, which shows that the CCC command applies to processors that have caches);

(b) detecting a write request to a shared memory location by a first debug session (see, for example, column 9, lines 1-11, which shows detecting an access request to a shared memory location, and column 8, lines 7-11, which shows that such requests include write requests);

(c) if the first processor associated with the first debug session has write access to the shared memory location then selecting the first processor to perform the write request (see, for example, column 9, lines 42-47, which shows determining the processor that has access to the shared memory location, and column 14, lines 22-26, which then shows selecting the first processor to perform the write request), else performing the following steps:

(i) searching the software memory map for a second processor with write access to the shared memory location (see, for example, column 9, lines 32-41, which shows determining if a "remote" or second processor has access to the shared memory location based on the memory map);

(ii) selecting the second processor to perform the write request (see, for example, column 15, lines 46-52, which shows selecting the second processor to perform the write request);

(d) passing the write request initiated by the first debug session to the selected processor for execution (see, for example, column 14, lines 22-26 and column 15, lines 46-52, which shows passing the write request to the selected processor);

(e) searching the software memory map for a second plurality of processors that have read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location);

(f) broadcasting the write request to the second plurality of processors (see, for example, column 10, line 66 to column 11, line 5, which shows broadcasting the request to the plurality of processors); and

(g) performing cache coherency updates in response to the write request in each of the second plurality of processors (see, for example, column 6, lines 46-53, which shows issuing and executing "CCC" commands, and column 3, lines 5-11, which shows that the CCC commands are for performing cache coherency updates).

With respect to claim 17, Tarui further discloses the limitation wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request (see, for example, column 14, line 5-14, which shows broadcasting an "I" command, and column 8, lines 7-11, which shows that the I command is for maintaining cache coherency as opposed to a normal write request).

With respect to claim 18, Tarui further discloses the limitation wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to

the write request intended for maintaining cache coherency (see, for example, column 8, lines 7-11, which shows using a cache coherency capability in response to a write request).

With respect to claim 19, Falik also discloses a digital system (see, for example, FIG. 21), comprising:

(a) multiple processors with common shared memory for executing an application program (see, for example, column 1, lines 34-49, and column 17, lines 46-48); and

(b) wherein the application program was developed with the software development system of claim 1 (see, for example, column 1, lines 46-49, and see claim 1 above).

11. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Falik in view of Tarui, as applied to claim 3 above, and further in view of U.S. Pat. No. 6,539,500 to Kahle et al. ("Kahle").

With respect to claim 6, Falik in view of Tarui does not expressly disclose the limitations wherein:

(a) the step of creating comprises denoting in the software memory map the shared memory locations that contain program instructions;

(b) the step of passing the write request additionally comprises the step of determining that the shared memory location contains a program instruction; and

(c) the cache is an instruction cache.

However, Kahle discloses instruction tracing in a shared memory multiprocessor system (see, for example, the abstract). Instructions can be traced and analyzed simultaneously with

instruction execution at the normal system speed (see, for example, column 2, lines 5-20), so as to decrease the time needed for debugging (see, for example, column 1, lines 45-55).

Kahle further discloses defining a memory map of the address space of program instructions in shared memory (see, for example, column 3, lines 19-27), as in part (a) above.

Kahle further discloses determining that the shared memory location contains a program instruction (see, for example, column 3, lines 48-50), as in part (b) above.

Kahle further discloses an instruction cache (see, for example, step 400 in FIG. 4), as in part (c) above.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the software development system of Falik and Tarui with the instruction tracing features taught by Kahle. One of ordinary skill in the art would have been motivated to provide such features so as to decrease the time needed for debugging.

12. Claims 9-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Falik in view of Tarui, as applied to claim 7 above, and further in view of U.S. Pat. No. 6,681,384 to Bates et al. ("Bates").

With respect to claim 9, Falik also discloses the step of running after hitting a breakpoint in a shared memory location (see, for example, column 17, lines 37-48).

Falik in view of Tarui does not expressly disclose the step of stepping over a software breakpoint.

However, Bates discloses debugging a multithreaded program and stepping the threads that are halted at a breakpoint (see, for example, the abstract). The threads are synchronized so

as to ease the burden of debugging multithreaded applications (see, for example, column 2, lines 56-58), which execute on multiprocessor systems (see, for example, column 1, lines 50-54).

Bates further discloses:

(a) requesting that a software breakpoint in a shared memory location be stepped over or program execution resumed after hitting the breakpoint in a third debug session (see, for example, column 8, lines 60-65, which shows requesting that a breakpoint be stepped over, and column 9, lines 13-19, which shows requesting that program execution be resumed);

(b) clearing the software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the clearing of the breakpoint (see, for example, column 8, lines 53-59, which shows deleting or clearing the breakpoint such that the debug session of each thread or processor is notified);

(c) stepping a processor associated with the third debug session to the instruction after the shared memory location from which the software breakpoint was cleared (see, for example, column 11, lines 5-14, which shows stepping to an instruction after the breakpoint); and

(d) setting the first software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the setting of the breakpoint (see, for example, column 8, lines 43-52, which shows setting the breakpoint such that the debug session of each thread or processor is notified).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the software development system of Falik and Tarui with the breakpoint features taught by Bates. One of ordinary skill in the art would have been motivated to provide such features so as to ease the burden of debugging multiprocessor systems.

With respect to claim 10, Tarui further discloses the step of:

(a) searching the software memory map for a sixth plurality of processors having read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location).

Bates further discloses:

(b) halting all processors in the sixth plurality of processors after the step of requesting and prior to the step of clearing (see, for example, column 10, line 61 to column 11, line 4, which shows suspending or halting the plurality of threads or processors).

With respect to claim 11, Tarui further discloses the limitation wherein the step of setting comprises:

(a) searching the software memory map to find a seventh plurality of processors having read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location).

Bates further discloses:

(b) updating a software representation maintained for software breakpoints for each of the seventh plurality of processors (see, for example, column 14, lines 34-35, which shows updating a breakpoint table); and

(c) writing the software breakpoint instruction in the shared memory location (see, for example, column 14, lines 40-41, which shows writing the breakpoint instruction).

With respect to claim 12, Bates further discloses the limitation wherein the step of clearing comprises:

(a) writing the original instruction stored in a software representation maintained for software breakpoints in the shared memory location (see, for example, column 14, lines 37-39, which shows writing the original instruction).

Tarui further discloses:

(b) searching the software memory map to find an eighth plurality of processors having read access to the shared memory location (see, for example, column 6, line 66 to column 7, line 9, which shows checking or searching a memory table or map for a plurality of processors that have access to the shared memory location).

Bates further discloses:

(c) updating a software representation maintained for software breakpoints for each of the eighth plurality of processors to remove the software breakpoint for the shared memory location (see, for example, column 8, lines 57-59, which shows updating a breakpoint table to remove the breakpoint).

Conclusion

13. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure. U.S. Pat. No. 6,718,294 to Bortfeld discloses a system and method for synchronized control of system simulators with multiple processor cores.

14. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER